

Assignment 10-1 Identify identifiers of C++ programs 1

An identifier is a series of characters consisting of letters, digits and underscores (`_`) that does not begin with a digit. You are given a syntactically correct C++ program, and you'll have to find out all non-keyword identifiers.

Input

The input consists of exactly one line with a file name. The corresponding file which contains a syntactically correct C++ program. For the sake of simplicity, we suppose that, in this cpp file, all comments are *single-line comments* which begin with `//`.

Output

You are to output a text file which consists of all non-keyword identifiers in the inputted cpp file.

Sample Input (the content of a cpp file)

test1.cpp

The contents of the file test1.cpp is as follows:

```
#include <iostream>
using namespace std;

int main()
{
    char s1[ 20 ];
    char s2[] = "happy new year";

    cout << "Enter the string \"merry christmas\": ";
    cin >> s1; // reads "merry"

    cout << "s1 is: " << '\'' << s1 << '\'' << "\ns2 is: " << '\'' << s2 << '\'';

    cout << "\n\ns1 with spaces between characters is:\n";
    for ( int i = 0; s1[ i ] != '\0'; i++ )
        cout << s1[ i ] << ' ';
    cout << '\n';

    cin >> s1; // reads "christmas"
    cout << "\ns1 is: " << '\'' << s1 << '\'' << '\n';
}
```

Sample Output (the contents of a text file identifiers.txt)

```
iostream
std
main
s1
s2
cout
cin
```

Part of the program

You are required to write the functions `load`, `delStrConsts`, `delCharConsts`, `extractIdentifiers` and `store` to complete the following program which solves this problem. In your program, you cannot declare global variables except keywords.

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace::std;

// reads in a C++ program from a cpp file, and put it to the array program
void load( char( *program )[ 100 ], int &numLines );

// deletes the comment beginning with "//" from sourceLine if any
void delComment( char sourceLine[] );

// deletes all string constants from sourceLine
void delStrConsts( char sourceLine[] );

// deletes all character constants from sourceLine
void delCharConsts( char sourceLine[] );

// extracts all identifiers from sourceLine, and
// put them into the array identifiers
void extractIdentifiers( char sourceLine[], char identifiers[][ 32 ],
                        int &numIdentifiers );

// stores all non-keyword strings in the array identifiers into a text file
void store( char ( *identifiers )[ 32 ], int numIdentifiers );

// returns true if and only if str is a C++ keyword
bool keyword( char str[] );

// returns true iff identifiers[ pos ] belongs to identifiers[ 0 .. pos-1 ]
bool duplicate( char ( *identifiers )[ 32 ], int pos );

const char keywords[][ 20 ] = { "auto", "break", "case", "char", "const",
                                "continue", "default", "define", "do", "double",
                                "else", "enum", "extern", "float", "for",
                                "goto", "if", "int", "long", "register",
                                "return", "short", "signed", "sizeof",
                                "static", "struct", "switch", "typedef",
                                "union", "unsigned", "void", "volatile",
                                "while", "bool", "catch", "class",
                                "const_cast", "delete", "dynamic_cast",
                                "explicit", "false", "friend", "inline",
                                "mutable", "namespace", "new", "operator",
                                "private", "protected", "public",
                                "reinterpret_cast", "static_cast", "template",
                                "this", "throw", "true", "try", "typeid",
                                "typename", "using", "virtual", "include" };

int main()
{
    char ( *program )[ 120 ] = new char[ 500 ][ 120 ];
    int numLines = 0;

    // reads in a C++ program from a cpp file, and put it to the array program
    load( program, numLines );

    char ( *identifiers )[ 32 ] = new char[ 500 ][ 32 ];
    int numIdentifiers = 0;

    for( int i = 0; i < numLines; i++ )
    {
```

```

        delComment( program[ i ] ); // deletes the comment beginning with "//"
    from program[ i ]
    delStrConsts( program[ i ] ); // deletes all string constants from
    program[ i ]
    delCharConsts( program[ i ] ); // deletes all character constants from
    program[ i ]

    if( strcmp( program[ i ], "" ) != 0 )
        extractIdentifiers( program[ i ], identifiers, numIdentifiers );
        // extracts all identifiers from program[ i ], and
        // put them into the array identifiers
    }

    // stores all non-keyword strings in the array identifiers into a text file
    store( identifiers, numIdentifiers );

    delete[] program;
    delete[] identifiers;
}

void load( char( *program )[ 120 ], int &numLines )
{

}

void delComment( char sourceLine[] )
{
    size_t length = strlen( sourceLine );
    if( length > 1 )
        for( size_t i = 0; i < length - 1; i++ )
            if( sourceLine[ i ] == '/' && sourceLine[ i + 1 ] == '/' )
                {
                    sourceLine[ i ] = '\0';
                    return;
                }
}

void delStrConsts( char sourceLine[] )
{

}

void delCharConsts( char sourceLine[] )
{

}

void extractIdentifiers( char sourceLine[], char identifiers[][ 32 ], int
&numIdentifiers )
{

}

void store( char ( *identifiers )[ 32 ], int numIdentifiers )
{

```

```

}

bool keyword( char str[] )
{
    size_t numKeywords = sizeof( keywords ) / 20;
    for( size_t i = 0; i < numKeywords; i++ )
        if( strcmp( keywords[ i ], str ) == 0 )
            return true;

    return false;
}

bool duplicate( char ( *identifiers )[ 32 ], int pos )
{
    for( int i = 0; i < pos; i++ )
        if( strcmp( identifiers[ i ], identifiers[ pos ] ) == 0 )
            return true;

    return false;
}

```